Reducing Cross-group Traffic with a Cooperative Streaming Architecture

Zhijie Shen
School of Computing
National University of Singapaore
Singapore 117417
z-shen@comp.nus.edu.sg

Roger Zimmermann
School of Computing
National University of Singapaore
Singapore 117417
rogerz@comp.nus.edu.sg

ABSTRACT

Cooperative approaches, such as P2P networks, have demonstrated their effectiveness in video delivery. However, with underlay structure considered, it is still possible to further improve traffic efficiency. In this paper, we discuss the problem of localizing the traffic traversal across peer groups, which are partitioned according to underlay characteristics. We first provide three concrete examples to demonstrate this common challenge, which we theoretically formulate afterwards. Finally, we propose a ring overlay approach, which performs excellently to solve the problem, while tolerating peer dynamics and supporting peer heterogeneity.

Categories and Subject Descriptors

C.2.4 [**Computer-Communication Networks**]: Distributed Systems

Keywords

Cooperative networks, peer-to-peer, wireless, traffic locality

1. INTRODUCTION

Cooperative paradigms, especially P2P networks, have demonstrated their significant capability of scaling video streaming applications in support of huge video corpora and audiences. "Last mile" connections were traditionally assumed to be the bottleneck of P2P networks. However, by studying the underlay structure, it was recently found that the bottleneck may also be the links between some "islands" of peers while there exists a relatively good connectivity within them. The islands are the sub-networks partitioned according to some underlay characteristics. For instance, they can be local area networks (LANs), Internet service provider (ISP) networks or WiFi connected mobile handsets. We term such an island of peers a *group*, the traffic transmitted within an island the *intra-group traffic* and the traffic transmitted between islands the *cross-group traffic* (see Figure 1). Hence to relieve bottleneck links at the group boundaries, cross-group traffic is desired to be reduced. Three examples are briefly discussed below.

CROSS-ISP TRAFFIC REDUCTION: A classic problem is reducing cross-ISP traffic. This traffic may raise the operational costs of ISPs and congest the gateways among them, which motivates researchers to investigate how to reduce it. It is found that part of this traffic is not necessary to keep video streams delivered in time, but is caused by inefficient overlay routing protocols (*e.g.*, random peer selection). Several solutions [8, 7, 12, 10] have been proposed to localize the unnecessary part. The common objective is to improve the utilization of the bandwidth resources from local peers, that is, to replace cross-ISP traffic with intra-ISP traffic.

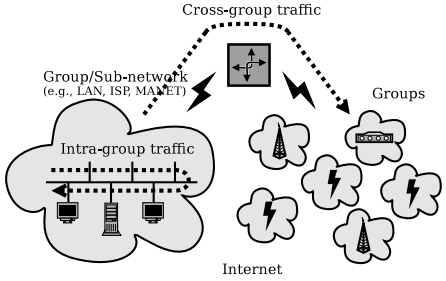

Figure 1: Partitioned P2P network structure.

LAN-SCALE TRAFFIC LOCALITY: A trend is that households own more digital products, which are connected to a home LAN, and access the Internet via a shared link. Recent measurements on PPTV [9] revealed that a noticeable number of LANs accommodated multiple peers watching the same TV channel. Surprisingly, up to 21% peers belonged to such LANs. Moreover, LAN access is usually fast and usually exempt from subscription fees, while Internet access is relatively slow and fee-based. Therefore, by being aware of the partners in the same LAN, peers can avoid requesting duplicate data remotely, saving data downloads from the Internet.

MOBILE WIRELESS NETWORKS: Extending the P2P paradigm to the booming mobile video market is desired, but not trivial due to the particular wireless network characteristics and the limited capacity of mobile handsets. One prominent challenge is that the intensive downloading and uploading through 3G networks exhausts data quotas and energy quickly. However, if close-by handsets form a mobile ad-hoc network (MANET) via the secondary network interface (*e.g.*, WiFi and Bluetooth), and use it to swap streaming data, part of the traffic will be off-loaded from 3G networks to the MANET [4, 5]. This is beneficial because the secondary network interface is cost free and consumes less energy. A further requirement here is load balance, since cooperators want to be treated fairly in spending their data quotas and energy.

There are some *common characteristics* of the networks among these three examples: 1) Video consumers in a certain group need to access video streams originating from outside the group. However, cross-group downloading is expensive in terms of network congestion, monetary cost, energy consumption, *etc.* 2) In contrast, the consumers can connect to the others in the same group more easily, and intra-group data transmission is relatively cheaper.

By observing common characteristics, we are inspired to devise

MM'12, October 29–November 2, 2012, Nara, Japan.

a compatible solution for all scenarios instead of a separate solution for each. Therefore this paper introduces two major contributions. First, we introduce three scenarios where peers desire to cooperatively import video streams to reduce cross-group traffic, that is, achieve ISP-scale and LAN-scale traffic locality and cooperative streaming among mobile devices. Second, we describe the common problem, and develop a linear program to formally describe it. We propose a ring overlay approach, which is an excellent solution of the linear program, while tolerating peer dynamics and supporting peer heterogeneity. Particularly, our approach is well suitable for mobile wireless networks thanks to its capability of performing workload balancing. In the the rest of the paper, we discuss our work in detail.

2. PROBLEM FORMULATION

To optimize cross-group traffic in the three examples, the general strategy is that video consumers cooperatively pull streaming data into their group and share it among themselves. This is a two-step procedure. We focus on the step of cooperatively importing video streams in this paper, since it determines how effectively cross-group traffic is eliminated. Our target is to keep consumers downloading as few as possible duplicate video streams from outside the group. The optimal result is that one and only one copy of the video streams is imported.

The cooperative downloading problem can be defined as follows. The whole video is divided into a number of chunks of equal size, s_1, s_2, ..., s_n. k consumers (or peers) are assumed to simultaneously stay in the same group. To assign the tasks of importing the chunks from outside the group, we leverage a scoreboard

$$\begin{array}{c} \\ p_1 \\ p_2 \\ \vdots \\ p_k \end{array} \begin{array}{c} \begin{array}{cccc} s_1 & s_2 & \dots & s_n \end{array} \\ \begin{bmatrix} \clubsuit & \clubsuit & \dots & \clubsuit \\ \clubsuit & \clubsuit & \dots & \clubsuit \\ \vdots & \vdots & \ddots & \vdots \\ \clubsuit & \clubsuit & \dots & \clubsuit \end{bmatrix} \end{array}$$

where $\clubsuit$ need to be replaced with 0 or 1. If p_i is designated to download s_j, $\langle p_i, s_j \rangle = 1$. Otherwise, $\langle p_i, s_j \rangle = 0$. In addition, we define the function

$$sch(\vec{s_j}) = \begin{cases} 1, & \sum_{i=1}^{k} \langle p_i, s_j \rangle \geq 1 \\ 0, & \text{otherwise} \end{cases} \quad (1)$$

$sch(\vec{s_j}) = 1$ means that the j^{th} chunk is scheduled by at least one peer. Continuous playback is the essential requirement of streaming applications. To guarantee good streaming quality, the imported video streams should be as complete as possible. In other words, the sum $\sum_{j=1}^{n} sch(\vec{s_j})$ should be as close to n as possible. Conversely, our target is to keep $\sum_{i=1}^{k} \sum_{j=1}^{n} \langle p_i, s_j \rangle$ small to cut the cost of cross-group streams.

Furthermore, it is beneficial to distribute the workload of downloading streaming data from outside the group among local peers. One reason is to avoid a single or few failure points. The other is to balance the downloading cost. The second reason is particularly important in the third aforementioned scenario, because data quotas and energy are still scarce resources for mobile handsets. Ideally, each peer should download $\frac{n}{k}$ chunks. To measure workload balance, we typically use the standard deviation of the number of downloaded chunks by each peer, denoted by $\sigma[\sum_{j=1}^{n} \langle p, s_j \rangle]$. A smaller standard deviation indicates a better balance.

Finally, the aggregate download bandwidth of the link that connects a group into the remaining world is usually limited. For simplicity, we also quantize the bandwidth as the number of chunks that can be downloaded simultaneously, denoted by $\mathcal{C}$. Thus, the whole problem can be formulated as finding a solution for the following linear program

$$\begin{array}{ll} \textbf{min} & \alpha \sum_{h=1}^{n} sch(\vec{s_h}) + \\ & \beta \sum_{i=1}^{k} \sum_{j=1}^{n} \langle p_i, s_j \rangle + \\ & \gamma(\sigma[\sum_{j=1}^{n} \langle p, s_j \rangle]) \\ \textbf{s.t.} & \sum_{i=1}^{k} \sum_{j=1}^{n} \langle p_i, s_j \rangle \leq \mathcal{C}. \end{array} \quad (2)$$

where α, β and γ are the weights ($\alpha < 0$, $\beta > 0$ and $\gamma > 0$).

3. RING OVERLAY APPROACH

We introduce a practical solution, learned from token ring networks [1], to solve the problem outlined above. The general protocol of it is as follows. Peers in each group are organized in a virtual ring, having one direct predecessor and one direct successor. The server issues a token which records the sequence number of the last scheduled chunk, and pushes it to a random peer. Peers then pass the token along the ring. Whenever a peer obtains the token, it downloads one chunk from outside the group, increases the sequence number by one, and forwards the token to its successor. The protocol ensures that each chunk will be downloaded at most once from outside the group. Consequently, the second item in the target function of the linear program is optimized.

3.1 Asynchronous Token Forwarding

One design choice is whether chunk downloading and token forwarding should be synchronized or not. To arrive at a decision, we investigate the relation between the speed of gaining chunks and that of consuming chunks. The inequation must hold that the time to download chunks is no longer than the playback duration of the chunks. Suppose the token travels one round (n peers), stays for t_i^f time at each peer p_i, and forces it to download ξ chunks, each of which can be played for t^p time. Then, the inequation is

$$\sum_{i=1}^{n} t_i^f \leq \sum_{i=1}^{n} \xi_i t^p. \quad (3)$$

As each peer is scheduled to download exactly one chunk, Equation (3) can be simplified as $\sum_{i=1}^{n} t_i^f \leq n t^p$. In the synchronized case, the token stays at a peer until the chunk is downloaded, such that $t_i^f = \frac{sizeof(s_i)}{c_i}$ where c_i is the download bandwidth of p_i. Similarly, $t^p = \frac{sizeof(s_i)}{r}$ where r is the streaming rate of the video. Then, the formula can be further modified to $\sum_{i=1}^{n} \frac{1}{c_i} \leq n\frac{1}{r}$, in which the right part is usually constant. If peers' bandwidth c_i is small enough, the inequality will not hold. Hence synchronization is not a sensible choice. Chunk downloading and token forwarding should be decoupled, that is, the token is free to move on when a peer only schedules, but yet not performs a downloading task. For simplicity, the token is forwarded at a constant interval t^f, thus Equation (3) can be simplified to $t^f \leq t^p$, indicating that the token forwarding interval must be no longer than a chunk's playback duration.

3.2 Multiple Successors

Traditionally, token ring networks [1] provide only one path between any two nodes, such that the networks are at risk of being broken when node failures occur. In the scenario of video streaming, the ring overlay is more vulnerable to network partitioning due to the nature of peer dynamics. To enhance the reliability of the overlay, each peer is supposed to connect to more than one peer. However, unlike Chord [11] where a peer maintains multiple outgoing links to the 2^{i-1}-th following peers to accelerate content lookup, a peer connects to its m immediate successors in our ring

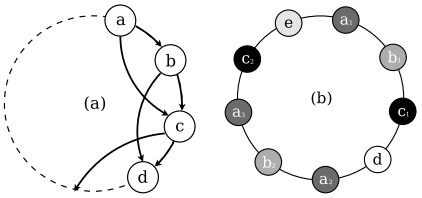

Figure 2: Illustration of (a) the ring overlay and (b) peers' multiple positions in the ring overlay.

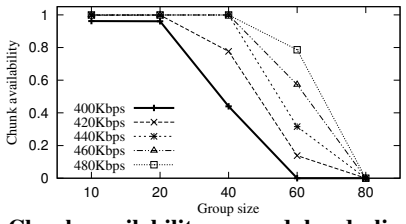

Figure 3: Chunk availability around borderline aggregate download bandwidth.

overlay. Figure 2(a) illustrates a sample ring overlay where $m = 2$. Now when a peer is to forward the token, it consecutively targets from the closest to the farthest successor until one acknowledges its reception. If no successor is reachable, the token will be forwarded to the sender itself. This protocol ensures that the least number of peers are skipped. Furthermore, multiple connections enable the ring overlay to repair itself with local information when less than m adjacent peers leave simultaneously. That is, a peer needs to just contact $O(m)$ others to recover connections. However, when m or more adjacent peers leave together, the peers at the two opposite sides of the gap cannot contact each other. In this case, the server must resort to rescuing the ring overlay with its global knowledge. Fortunately, even a small m makes this situation highly unlikely, since the occurrence rate drops exponentially while m increases.

3.3 Token Loss Detection

One more issue in token ring networks is token loss. In our approach, token loss happens when a peer who keeps the token leaves before it forwards the token to its successor. Therefore, whenever the server observes the leave of a peer, it needs to check whether the token was held by this peer. If it is lost, a new token has to be issued and inserted into the ring overlay. The process will be complex when a peer leaves abnormally (or crashes). Since the peer cannot report to the server, the server has to investigate the ring overlay to confirm whether the token is lost or not. However, this is generally a rare case.

4. EXPERIMENTAL EVALUATION

To evaluate our approach, we implemented a prototype, and configured it as follows. Peers download a 400 Kbps video stream. The video stream is divided into a sequence of 200 KB chunks, such that each chunk can be played for 4 seconds. The token is forwarded every 3 seconds, in accordance with Equation (3). Below we discuss the experimental results obtained from our simulations.

4.1 Completeness

First, how complete the imported video is in the group is of the highest importance. To measure completeness, we define a term, *chunk availability*, which is the fraction of requested chunks that has been received by any peer in the group before their playback deadline. We evaluated the chunk availability with different download bandwidth settings. When having plentiful bandwidth resources, the chunk availability is always very close to 1 (above

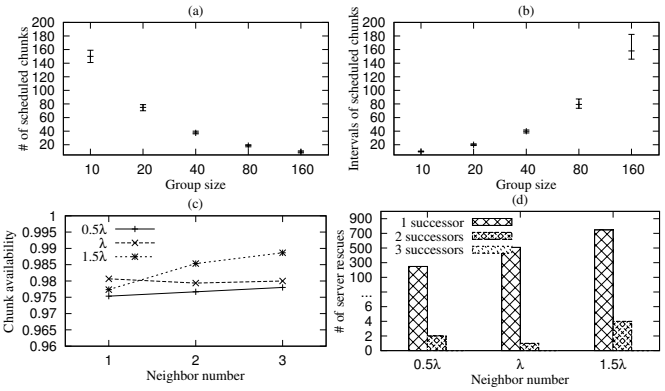

Figure 4: (a) Number of the chunks that are scheduled by the peers and (b) interval between 2 consequent scheduled chunks of each peer; relationship between (c) chunk availability / (d) number of server rescues and peers' neighbor cardinality.

0.97). The few chunk misses occur because they are scheduled by peers, but before they are downloaded, the peers have already left.

The more interesting scenario is when the aggregate download bandwidth of peers is only slightly higher than the streaming rate. Ideally, if the download bandwidth is no less than the streaming rate, the video can be downloaded uninterruptedly. However, the real situation is not that simple, as is shown in Figure 3. At the borderline where the total 400 Kbps download bandwidth is evenly shared by peers, the chunk availability cannot reach close to 1, and drops quickly. The reason is that as the number of peers grows, each peer is assigned an increasingly smaller share of the download bandwidth. Consequently, peers download chunks more slowly, such that before a peer fulfills a chunk downloading task, it is likely to have a next task ready to handle or to leave. In the extreme case where peers get too small share each, the chunk availability may decline to 0, because none of the chunks can be downloaded before their playback deadline. Hence the group size cannot freely scale.

4.2 Workload Balance

Next, we discuss how balanced chunk downloading tasks are distributed among peers. Figure 4(a) shows that the number of chunks scheduled by each peer is inversely proportional to the group size. Moreover, the deviation of chunk quantity is always tiny in contrast to the mean, indicating that each peer schedules a similar number of chunks. In addition, Figure 4(b) illustrates the average interval durations between any two consequent scheduled chunks on a peer and their deviation, which is still relatively small. It shows that peers schedule a new chunk downloading task at rather stable intervals. To summarize, our approach achieves both spatial and temporal balance of the distribution of chunk downloading tasks.

4.3 Peer Heterogeneity

In many real cases, it is required to keep the workload proportional to each peer's download capacity. Hence our protocol needs to be modified slightly to support peer heterogeneity. There are two choices. Option 1 is that when a peer receives the token, it schedules the number of chunks that is proportional to its download bandwidth. Option 2 is that inspired by the design of Dynamo [3], a peer is considered as a certain number of virtual peers according to its download bandwidth, and inserted into different positions in the ring overlay (*e.g.*, Figure 2(b)). The latter strategy should be better than the former, because peers under the former strategy are likely to schedule several chunks in bursts back-to-back.

Figures 5(a) and (b) illustrate the distributions of chunk down-

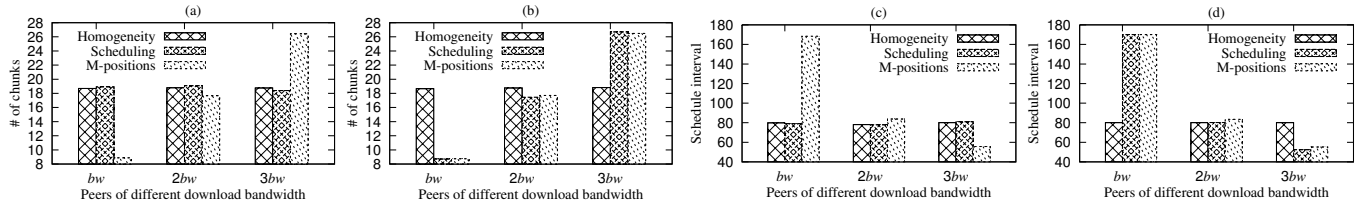

Figure 5: Number of the scheduled chunks of the peers with different download bandwidths in (a) live streaming and (b) VoD, and the interval of these chunks in (c) live streaming and (d) VoD.

loading tasks among the peers with three different download bandwidths in a live streaming and a VoD session. It is surprising that the former strategy cannot proportionally distribute the workload in the live streaming session (also seen in Figure 5(c)). By investigating the traces, we find that with live streaming, the chunks are generated just in time, such that peers having larger download bandwidth can rarely schedule multiple chunks. In contrast, peers are free to schedule multiple chunks in the VoD session since the chunks already exist. Fortunately, the latter strategy performs well in both cases. Additionally, Figure 5(d) confirms that the latter strategy promises a better temporal workload balance. Therefore, the latter strategy is preferred to cope with peer heterogeneity.

4.4 Churn

As peer dynamics occur naturally in P2P networks, it is interesting to know whether any extra connections to peers' successors are effective in alleviating the impact of peer dynamics. To make the simulations realistic, the arrival rate λ and the session length of the peers are modeled from our previous measurements [10].

From Figure 4(c), we observe that there is no obvious difference in chunk availability when the connection cardinality increases. This is reasonable because peer dynamics will influence the chunk scheduling only when the token is close to or at the position where a peer joins or leaves the ring overlay. In contrast, Figure 4(d) shows that the additional connections to peers' successors significantly relieve the server from the workload of rebuilding the broken ring overlay. When a peer only has one connection to its direct successor, the ring is not repairable given the leaving peer does not mediate the connection between its predecessor and successor. However, when a peer adds one more connection, the number of server rescues abruptly decreases because it is uncommon that two adjacent peers leave together, such that a peer can always recover one connection through the successor still alive. When there is a third connection for a peer, the number of server rescues decreases to 0. Hence a small connection cardinality can keep the ring overlay robust, though it does not help to improve chunk availability.

5. RELATED WORK

Cross-ISP traffic locality has been studied before. Picconi *et al.* [8] proposed a chunk scheduling strategy where peers request most data from its topologically-close partners and only resort to distant peers when data is not available locally. Magharei *et al.* [7] proposed a similarly two-tier scheduling scheme. Tomozei *et al.* [12] theoretically derived an intriguing, decentralized algorithm based on flow control, while we earlier presented a practical approach requiring joint efforts by peers and the server [10]. These prior approaches discussed traffic locality in the context of the wired Internet only, and did not consider balancing the workload of downloading video streams across group boundaries.

Some studies discussed cooperative video streaming approaches in mobile wireless networks. COSMOS is such an approach where a few peers download video streams through 3G networks and broadcast them to nearby neighbors through a no-cost network interface (*e.g.*, WiFi and Bluetooth) [4]. Compared with COSMOS, our approach can achieve a better workload balance. Furthermore, our approach is more general, such that it can be used in more than the scenario of mobile wireless networks. Liu *et al.* [5] proposed a similar approach but used burst streaming to save energy consumption. However, the assumption that every handset is reachable through broadcast narrows the scenarios to which this approach applies. On the contrary, our approach connects peers into an overlay. Additionally, there are studies that optimize mobile wireless streaming in other ways [2, 6].

6. CONCLUSIONS

We showed that with underlay structure exposed, traffic traversing certain kinds of network boundaries can effectively be optimized, and we supplied three scenarios. Afterwards, we analytically described the problem of localizing cross-group traffic, and proposed a ring overlay approach to solve it. The simulations confirmed the excellent performance of our approach.

Acknowledgment

This research has been supported by the Singapore National Research Foundation under its International Research Centre @ Singapore Funding Initiative and administered by the IDM Programme Office.

7. REFERENCES

[1] W. Bux, F. Closs, K. Kuemmerle, H. Keller, and H. Mueller. Architecture and Design of a Reliable Token-Ring Network. *IEEE JSAC*, 1983.

[2] J. Dai, B. Li, F. Liu, B. Li, and J. Liu. Collaborative Caching for Video Streaming among Selfish Wireless Service Providers. In *IEEE GLOBECOM*, 2011.

[3] D. Hastorun, M. Jampani, G. Kakulapati, A. Pilchin, S. Sivasubramanian, P. Vosshall, and W. Vogels. Dynamo: Amazon's Highly Available Key-value Store. In *ACM SOSP*, 2007.

[4] M.-F. Leung and S. H. G. Chan. Broadcast-Based Peer-to-Peer Collaborative Video Streaming Among Mobiles. *IEEE Transactions on Broadcasting*, 2007.

[5] Y. Liu and M. Hefeeda. Video Streaming over Cooperative Wireless Networks. In *ACM Multimedia Systems*, 2010.

[6] Y. Liu, F. Li, L. Guo, Y. Guo, and S. Chen. BlueStreaming: Towards Power-EfĩňĄcient Internet P2P Streaming to Mobile Devices. In *ACM Multimedia*, 2011.

[7] N. Magharei, R. Rejaie, V. Hilt, I. Rimac, and M. Hofmann. ISP-Friendly Live P2P Streaming. Technical report, University of Oregon, 2009.

[8] F. Picconi and L. Massoulie. ISP Friend or Foe? Making P2P Live Streaming ISP-Aware. In *IEEE ICDCS*, 2009.

[9] Z. Shen and R. Zimmermann. LAN-Awareness: Improved P2P Live Streaming. In *ACM NOSSDAV*, 2011.

[10] Z. Shen and R. Zimmermann. ISP-Friendly P2P Live Streaming: A Roadmap to Realization. *ACM TOMCCAP*, 2012.

[11] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications. *ACM SIGCOMM*, 2001.

[12] D.-C. Tomozei and L. Massoulie. Flow Control for Cost-Efficient Peer-to-Peer Streaming. In *IEEE INFOCOM*, 2010.